Andrew N6 = MACHINE LEARNING - Stanford

Linear algebra with one variable - regression (1)

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

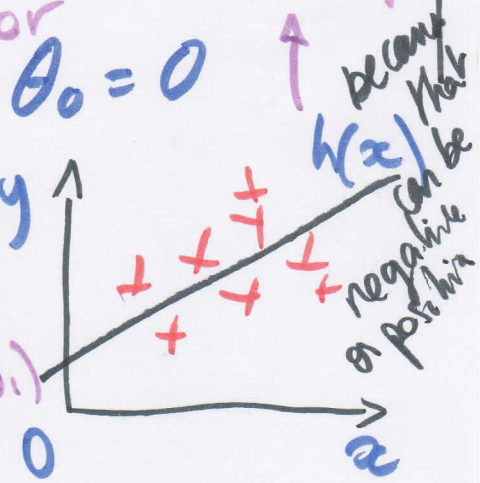
Cost function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$ → m = number of training examples

Goal = $\underset{\theta_0, \theta_1}{\text{Minimize}}\ J(\theta_0, \theta_1)$ → minimize the average error

Simplified: $h_\theta(x) = \theta_1 x$

$\theta_0 = 0$

$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$



Minimize $J(\theta_1)$ ← find the value θ_1 that minimizes $J(\theta_1)$
θ_1

Gradient Descent

Repeat until convergence $\theta_j := \theta_j - \alpha \underbrace{\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)}_{\text{derivative}}$

for $j = 0$ and $j = 1$

Gradient Descent for Linear Regression

Repeat until convergence { $\frac{\partial}{\partial \theta_0} J(\theta)$ $x_0^{(i)} = 1$

$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$

$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) x^{(i)})$ }

Update θ_0 and θ_1 simultaneously

Andrew Ng - MACHINE LEARNING - Stanford (2)

Gradient Descent for multiple variables

Hypothesis : $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$

Parameters : $\theta_0, \theta_1, \dots, \theta_n$ $\quad [\theta_0, \theta_1 \dots \theta_n] \leftarrow \theta^T$

Cost function : $J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Gradient descent : • Repeat { $\quad x_0^{(i)} = 1$

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \dots, \theta_n)$ }

(simultaneously update for every $j = 0, \dots, n$)

• Repeat {

$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$ }

(simultaneously update θ_j for $j = 0, \dots, n$)

Feature scaling

• Make sure features are on a similar scale

→ converge faster

ex : x_1 = size (0 - 2000 feet^2) $\quad x_2$ = nb of bedrooms (1-5)

$x_1 = \frac{\text{size (feet}^2\text{)}}{2000} \quad x_2 = \frac{\text{nb of bedrooms}}{5}$

• $0 \leq x_1 \leq 1 \quad 0 \leq x_2 \leq 1$

• Get every feature into approximately a $-1 \leq x_i \leq 1$ range

multiple features variables (ex)

Size	Nb of bed	Age	Price
2104	5	1	460
⋮	⋮	⋮	⋮

Mean normalization

- Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$)

ex. $x_1 = \frac{size - 1000}{2000}$ $\quad x_2 = \frac{\#bedrooms - 2}{5}$

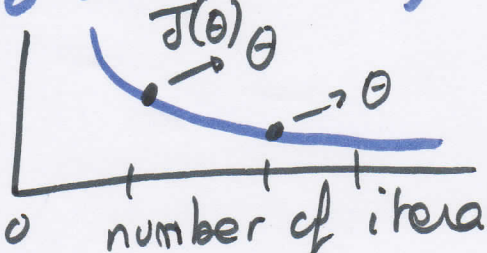
- $-0,5 \leq x_1 \leq 0,5$. $-0,5 \leq x_2 \leq 0,5$

$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1}$ — μ_1 → average value of x_1 in training set (mean); s_1 → range (max − min) or standard deviation

"Debugging": How to make sure gradient descent is working correctly

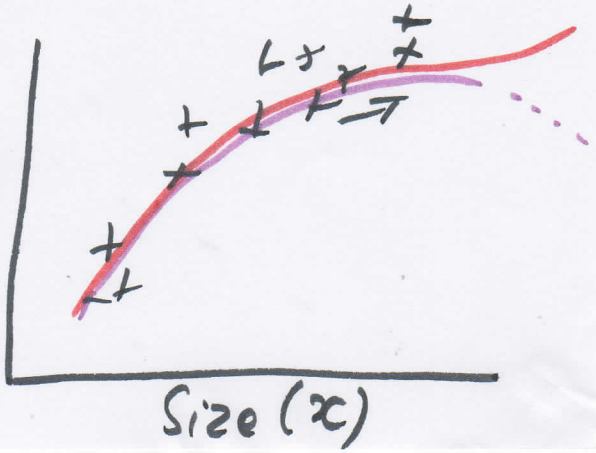
- For sufficiently small α, $J(\theta)$ should decrease on every iteration
- But if α is too small, gradient descent can be slow to converge
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge (slow convergence also possible)

To choose α, try ..., 0.001, 0.01, 0.1, 1, ...



Polynomial regression (1/2)

- $\theta_0 + \theta_1 x + \theta_2 x^2$.
- $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$.



Andrew Ng MACHINE LEARNING Stanford

Polynomial regression (2/2)

algorithms can help us find what polynomial regression to use, analyze the data ④

- $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 \sqrt{x}$

Normal equation

- m examples $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$; n features

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \quad x_0 = 1 \quad \in \mathbb{R}^{n+1}$$

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix} \quad m \text{ examples} \quad m \times (n+1) \quad n \text{ features}$$

X design matrix

e.g. if $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ \vdots & \\ 1 & x_1^{(m)} \end{bmatrix} \quad m \times 2 \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

- $\theta = (X^T X)^{-1} X^T y$ (inverse)

Octave: `pinv(X'*X)*X'*y`

do not work for some complex algorithms

- m training examples, n features

Gradient Descent	Normal Equation
- Need to choose α	- No need to choose α
- Needs many iterations	- Don't need to iterate
- Works well even when n is large	- Need to compute $(X^T X)^{-1}$
	- Slow if n is very large

at this point, it might be better to use gradient descent

$n = 10^6$

$n = 100, 1000, 10\,000$.

Andrew Ng - MACHINE LEARNING - Stanford

Logistic regression for classification

= sigmoid

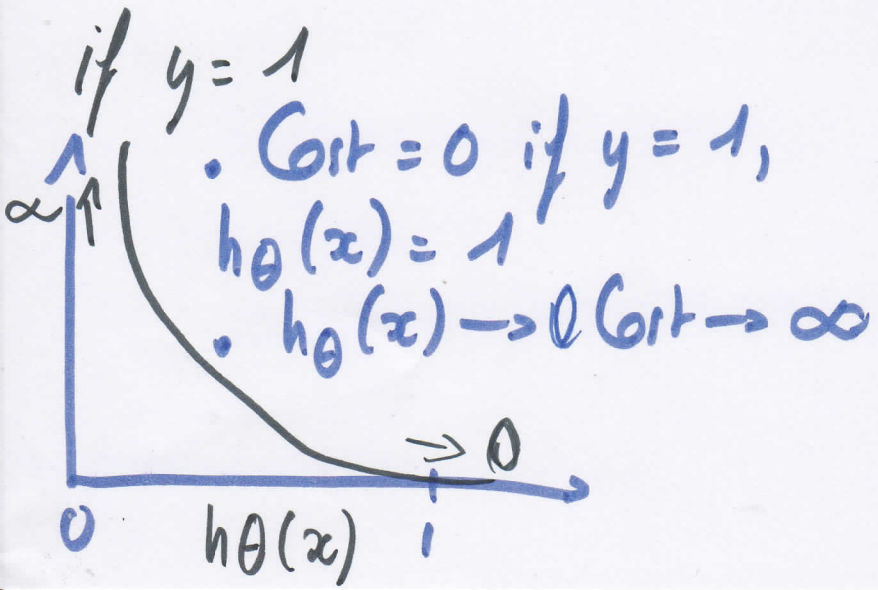
- $h_\theta(x) = g(\theta^T x)$
- $g(z) = \frac{1}{1+e^{-z}}$
- $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

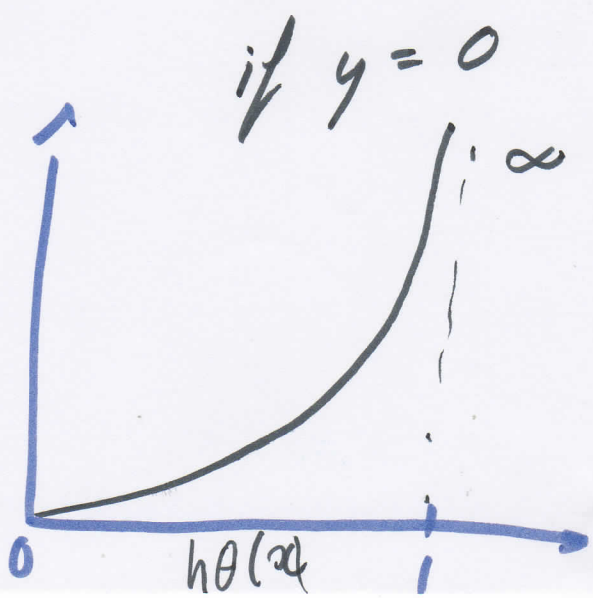
$0 \leq h_\theta(x) \leq 1$

- Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
- m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$ $x_0 = 1$, $y \in \{0, 1\}$ $\mathbb{R}^{n+1}$

Logistic regression cost function (1/2)

- $Cost(h_\theta(x^{(i)}), y^{(i)}) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$





Logistic regression cost function (2/2)

- $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$

Note: $y = 0$ or 1 always

- $J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) * \log(1-h_\theta(x^{(i)}))\right]$

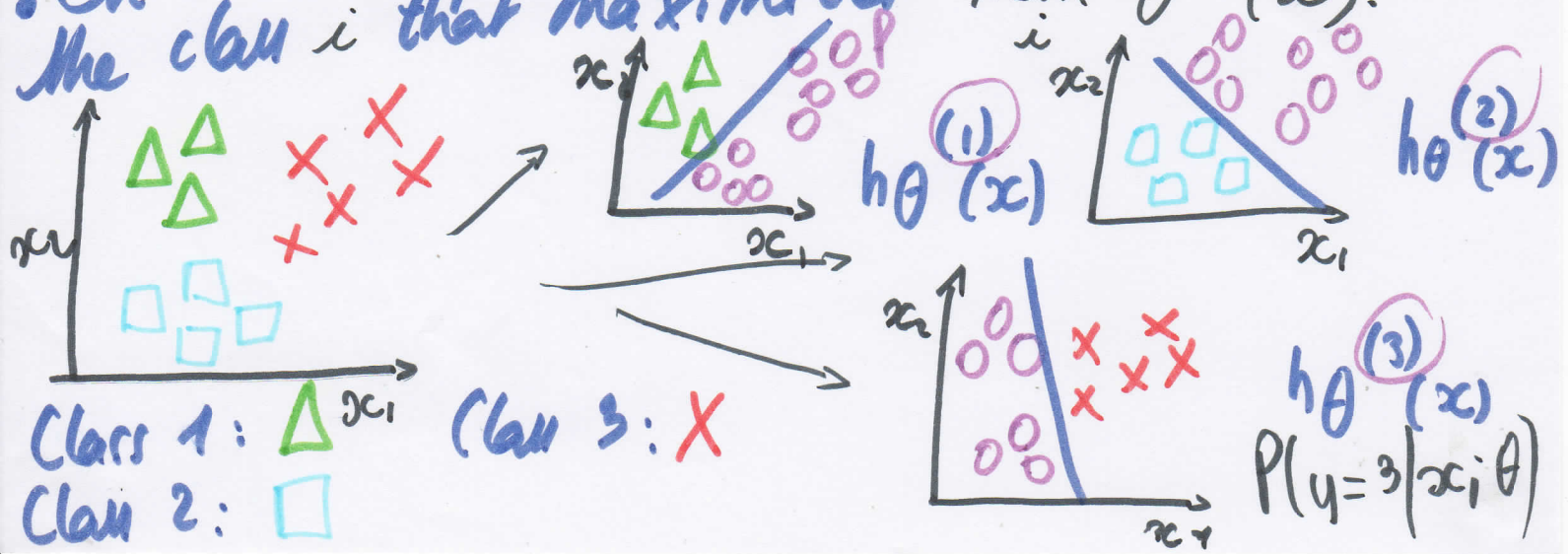
Want $\min_\theta J(\theta)$:

Repeat { $\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$ }

simultaneously update alle θ_j

Multiclass classification - one vs all

- Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.
- On a new input x, to make a prediction, pick the class i that maximizes $\max_i h_\theta^{(i)}(x)$.



Andrew Ng - MACHINE LEARNING - Stanford

Regularization - Regularized linear regression (7)

- $J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n}\theta_j^2\right]$

regularization term — parameter

- $\min_\theta J(\theta)$

θ_0 not penalized

It is important to choose well the regularization parameter λ

→ gradient descent

- $\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

↳ $\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$

$\rightarrow \min_\theta J(\theta)$

- Normal equation

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m$$

$m \times (n+1)$

if $\lambda > 0$

$$\theta = \left(X^TX + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \end{bmatrix}_{(n+1)\times(n+1)}\right)^{-1} X^Ty$$

- Regularized logistic regression

$$J(\theta) = \left[-\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log 1 - h_\theta(x^{(i)})\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

→ penalize $\theta_1, \theta_2, \ldots$ for being too large

Andrew Ng MACHINE LEARNING - Stanford

(8)

Supervised learning

Gives the "right answer" for each example in the data

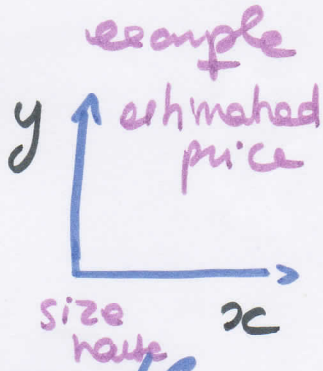
Regression problem

Predict real valued output

Classification

Discrete-valued output

Supervised learning - training set



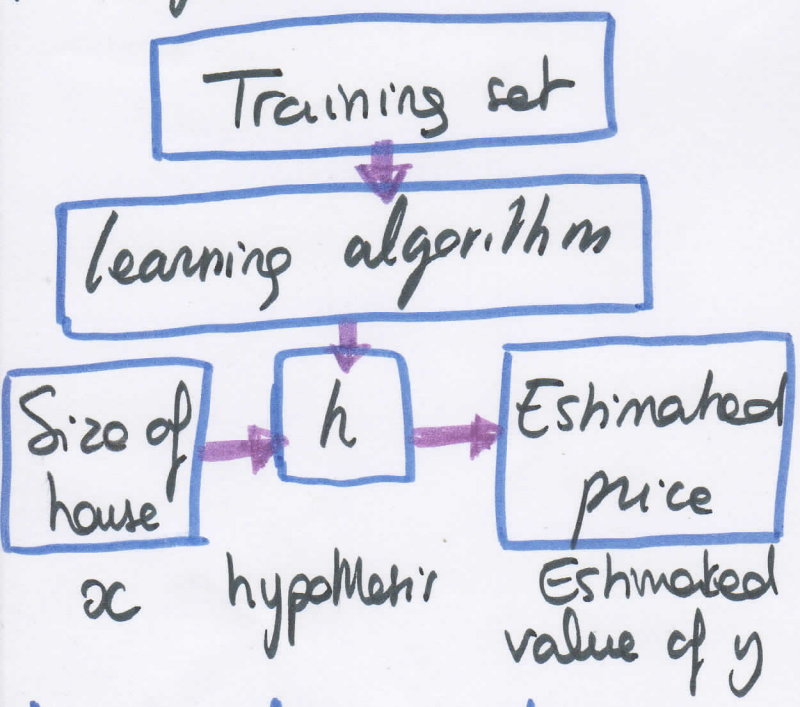
m = number of training examples

x's = "input" variable / features

y's = "output" variable / "target" variable

(x, y) = one training example

$(x^{(i)}, y^{(i)})$ = i^{th} training example



h maps from x's to y's

How do we represent h?

$h_\theta(x) = \theta_0 + \theta_1 x$

Shorthand: $h(x)$

Linear regression with one variable

Univariate linear regression

Lo one variable

Andrew Ng - MACHINE LEARNING - Stanford

Cost function

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$ - Parameters: θ_0 θ_1

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

Cost function: squared error function

$h(x)$ (for fixed θ_1, this is a function of x)

$\theta_1 = 1$

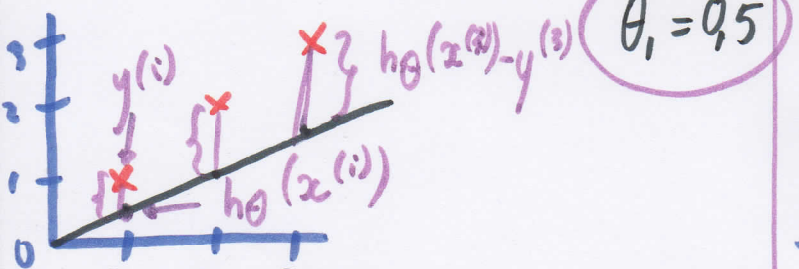
$J(1) = 0$

$h_\theta(x^{(i)}) = y^{(i)}$

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$= \frac{1}{2 \times 3} (0^2 + 0^2 + 0^2) = 0^2 = 0$$

$\theta_1 = 0.5$



$$J(0.5) = \frac{1}{2 \times 3} \left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

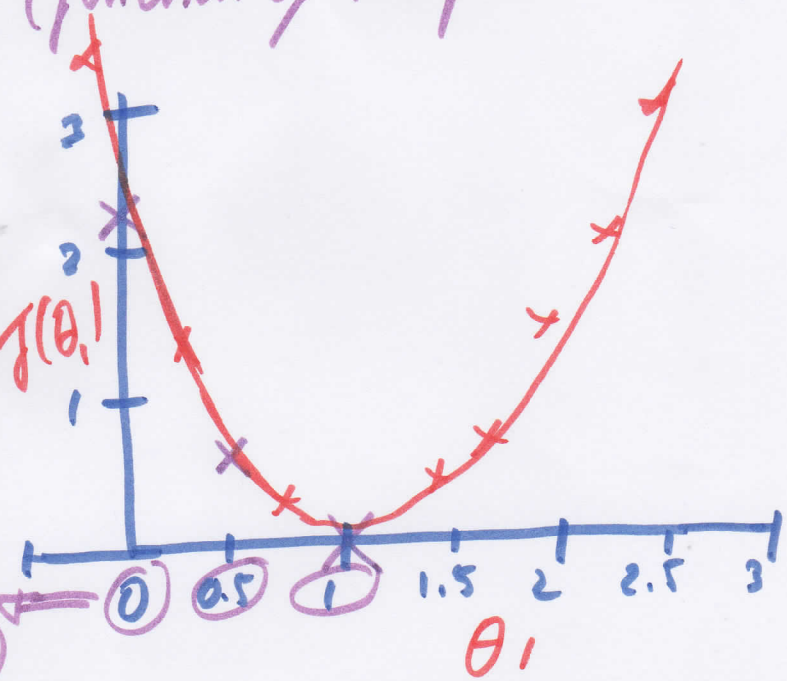
$$= \frac{3.5}{6} \simeq 0.58$$

$\theta_1 = 0$

$$J(0) = \frac{1}{6}(1^2 + 2^2 + 3^2)$$

$$= \frac{14}{6} \simeq 2.3$$

$J(\theta_1)$ (function of the parameter θ_1)

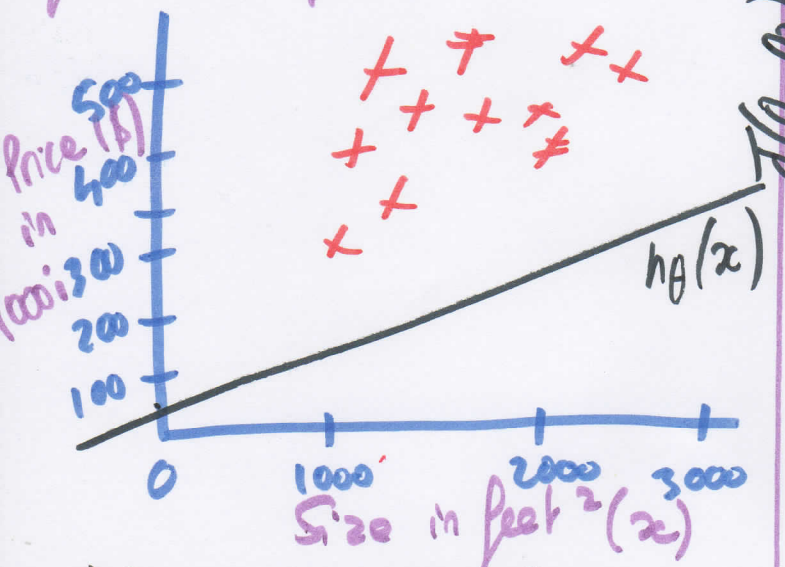


Each value of θ_1 corresponds to a hypothesis $h_{\theta_1}(x)$, which means finding a straight line that fits the data well.

Andrew Ng - MACHINE LEARNING - Stanford

$h_\theta(x)$

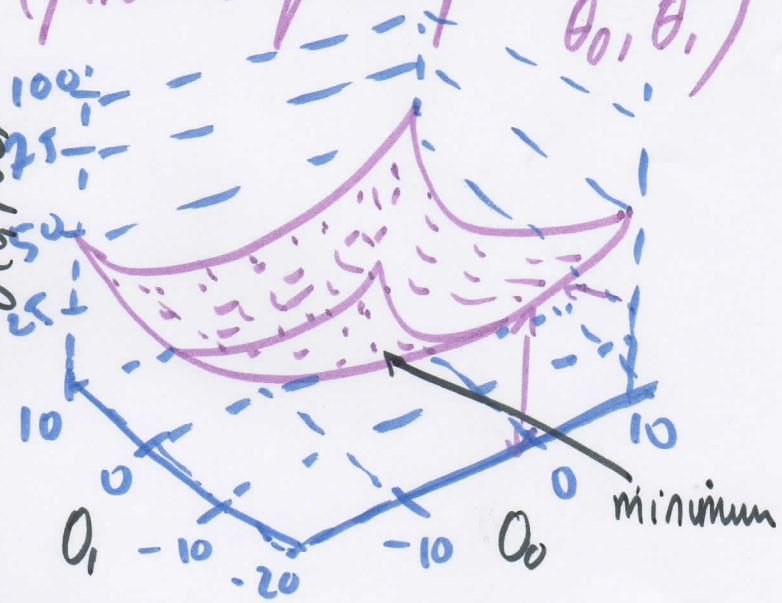
(for fixed θ_0, θ_1, this is a function of x)



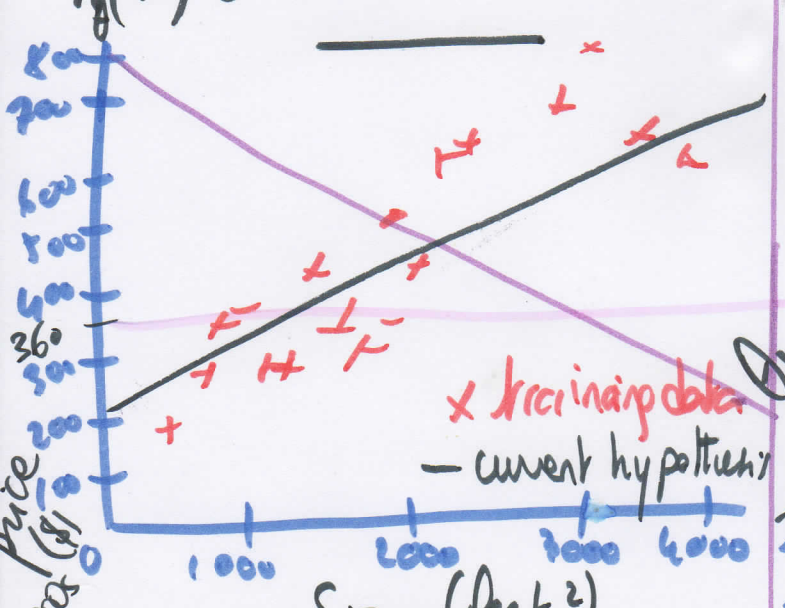
$h_\theta(x) = 50 + 0.06x$

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



contour plot

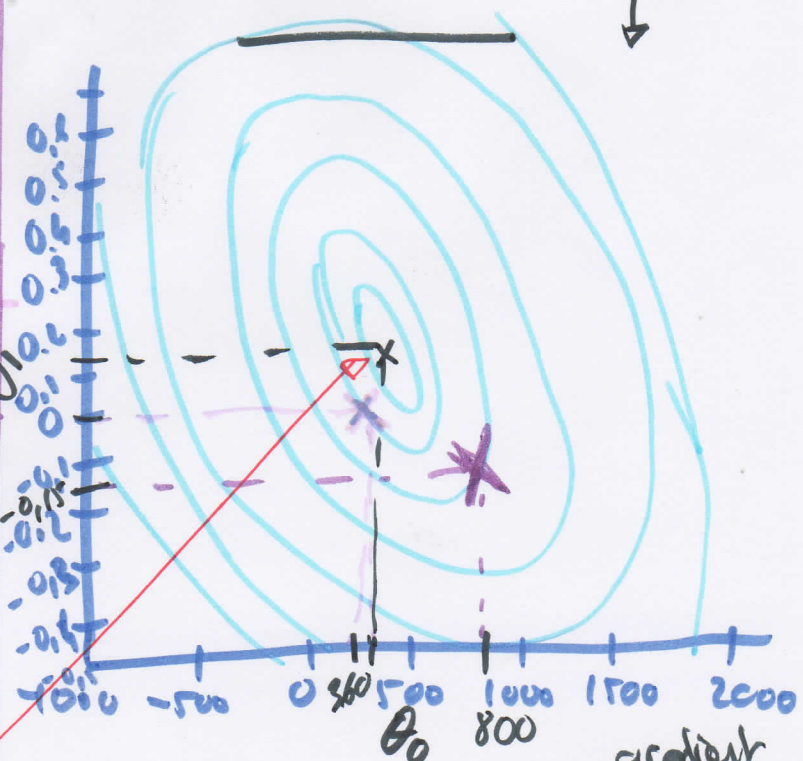


$h(x) = 800 - 0.15x$

$h(x) = 360 + 0x$

$h(x) = 220 + 0.15x$

fit better the training set



gradient descent

Outline

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

Andrew Ng - MACHINE LEARNING - Stanford

(11)

Gradient descent

- Minimize the cost function $J(\theta_0, \theta_1)$ and other functions
- Have some function $J(\theta_0, \theta_1)$ Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

as well with $J(\theta_0, \theta_1, \theta_2 \dots, \theta_n)$

- Outline:
  - ⇒ Start with some θ_0, θ_1
  - ⇒ Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum
- Repeat until convergence $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1) \}$$

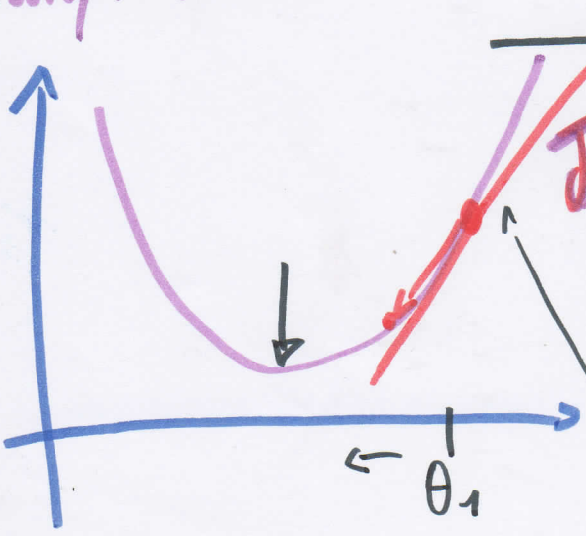
alpha = learning rate, it controls how big the step is when going down with gradient descent

if small = baby step
if large = huge step

Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

assignment



$(\theta_1 \in \mathbb{R})$

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

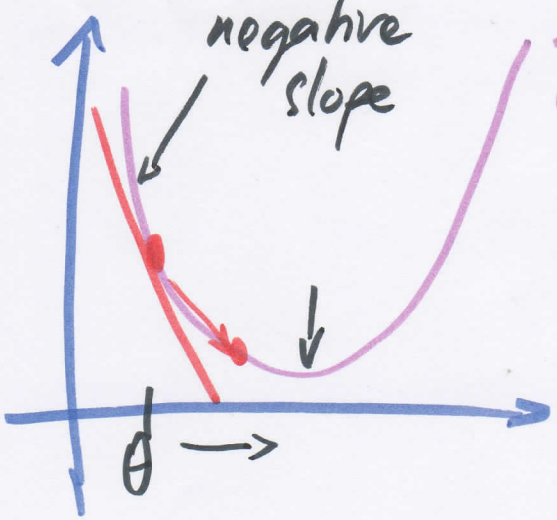
derivative ≥ 0 define slope

positive slope

$$\theta_1 := \theta_1 - \alpha \cdot (\text{positive number})$$

decreasing θ_1

Andrew Ng - MACHINE LEARNING - Stanford



$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1) \quad (2)$$

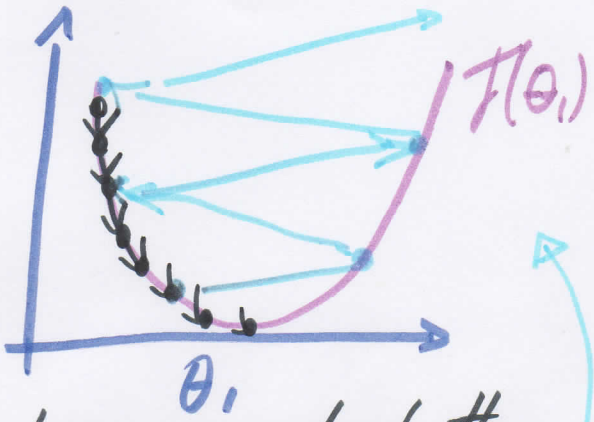
≤ 0

$\theta_1 := \theta_1 - \alpha \cdot$ (negative number)

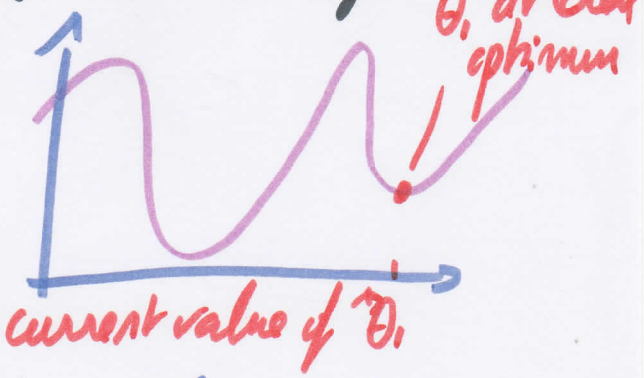
increasing θ_1

α too small or too large

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

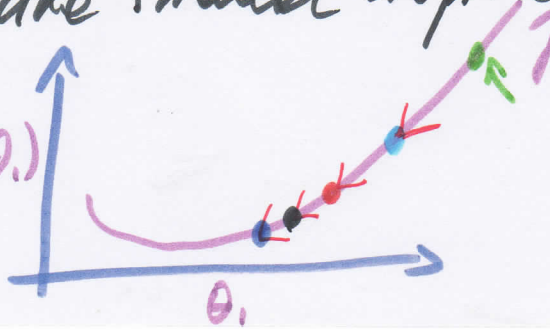


- If α is too small, gradient descent can be slow.
- if α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.
- If θ_1 is at a local optimum of $J(\theta_1)$, one step of gradient descent will leave θ_1 unchanged



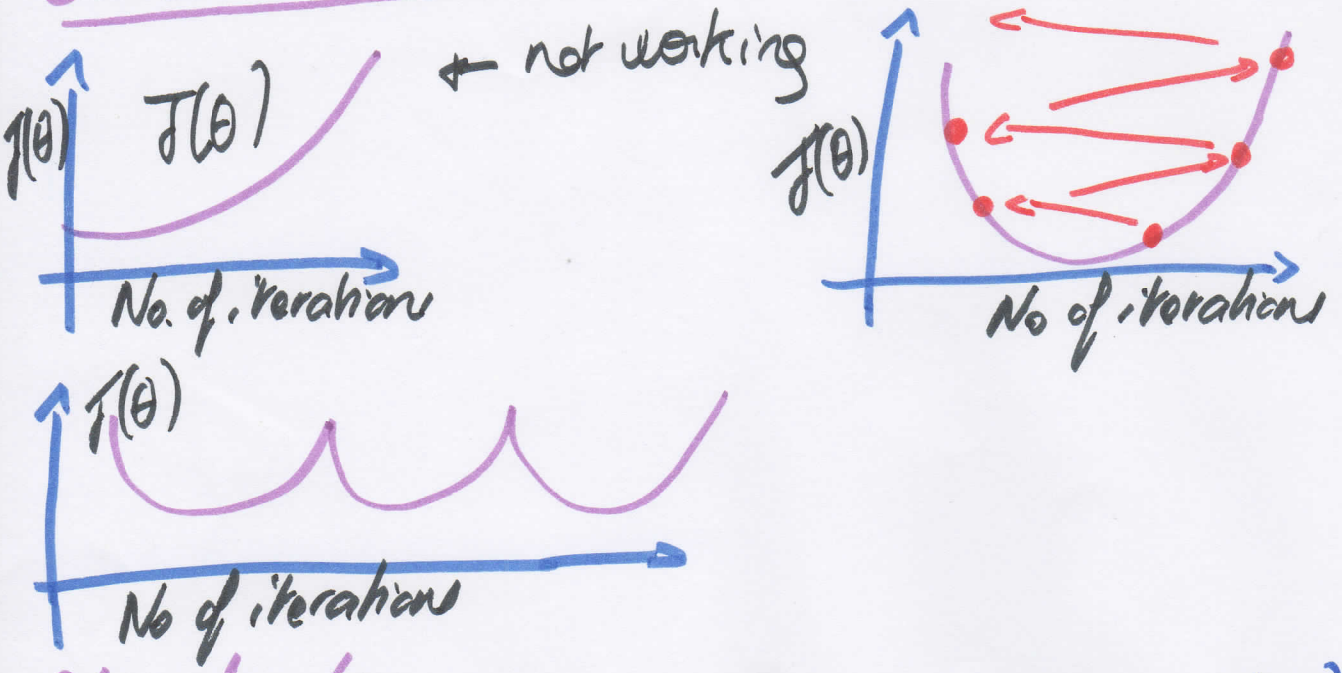
- Gradient descent can converge to a local minimum, even with the learning rate α fixed.

As we approach a local minimum, gradient descent will automatically take smaller steps. So no need to decrease α over time.



Andrew Ng - MACHINE LEARNING - Stanford (13)

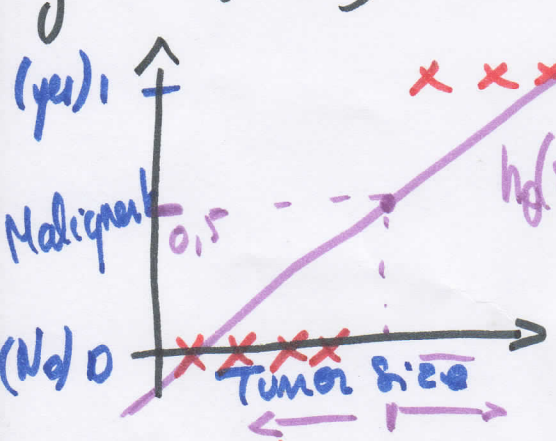
Gradient descent not working correctly



Classification

$y \in \{0, 1\}$

0: "Negative Class" (e.g. No)
1: "Positive Class" (e.g. Yes)



Threshold classifier output $h_\theta(x)$ at 0.5:

- If $h_\theta(x) \geq 0.5$, predict "$y = 1$"
- If $h_\theta(x) < 0.5$, predict "$y = 0$"

⚠ Classification: $y = 0$ or 1, however $h_\theta(x)$ can be > 1 / < 0
so: no linear regression but logistic regression
where: $0 \leq h_\theta(x) \leq 1$

Interpretation of Hypothesis Output with logistic regression

$h_\theta(x)$ = estimated probability that $y = 1$ on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumor size} \end{bmatrix}$ $h_\theta(x) = 0.7$

Tell patient that 70% chance of tumor being malignant

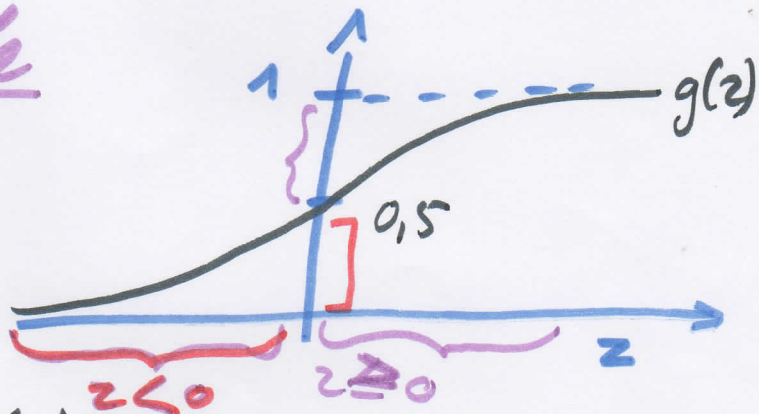
$h_\theta(x) = P(y=1|x;\theta)$ "probability that $y=1$, given x, parameterized by θ"

- $P(y=0|x;\theta) + P(y=1|x;\theta) = 1$
- $P(y=0|x;\theta) = 1 - P(y=1|x;\theta)$

Logistic regression example

$h_\theta(x) = g(\theta^T x)$;

$g(z) = \frac{1}{1+e^{-z}}$



Suppose predict "$y=1$" if $h_\theta(x) \geq 0,5$

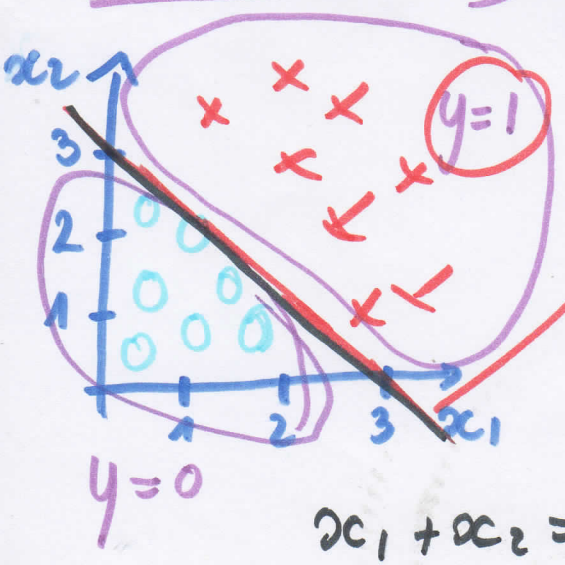
$g(z) \geq 0,5$ when $z \geq 0$ $h_\theta(x) = g(\theta^T x) \geq 0,5$

whenever $\theta^T x \geq 0$

predict "$y=0$" if $h_\theta(x) < 0,5$ $h_\theta(x) = g(\theta^T x)$ $g(z) < 0,5$

$\theta^T x < 0$

Decision boundary



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ $\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

Decision boundary

property of the hypothesis but for the dataset (~~in that case~~)

Predict "$y=1$" if $\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0$

$x_1 + x_2 \geq 3$

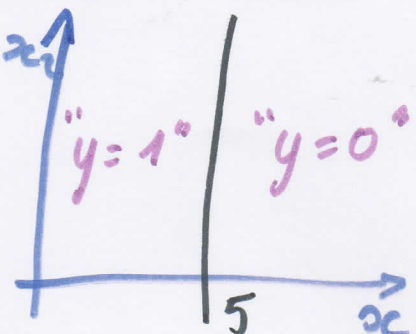
$x_1 + x_2 = 3$ $h_\theta(x) = 0,5$

$x_1 + x_2 < 3$ $y = 0$

Decision boundary

of $h_\theta(x) = g(5 - x_1)$

with $\theta_0 = 5, \theta_1 = -1, \theta_2 = 0$



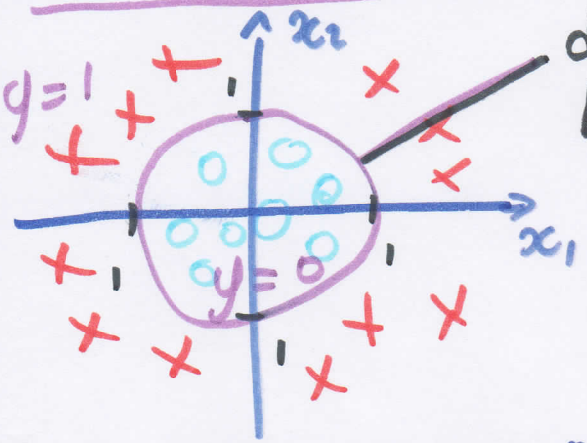
$y = 1$

$5 - x_1 \geq 0$

$-x_1 \geq -5$

$x_1 \leq 5$

Non linear decision boundaries



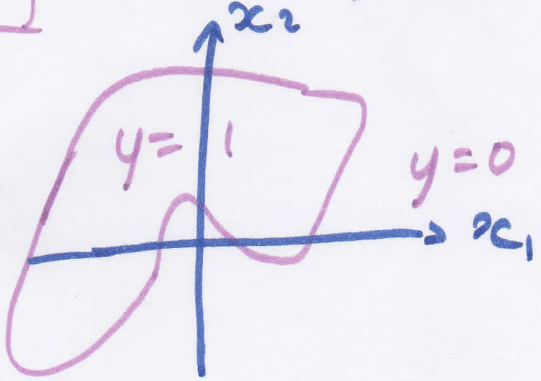
decision boundary

$h_\theta(x) = g(\underbrace{\theta_0}_{-1} + \underbrace{\theta_1 x_1}_{0} + \underbrace{\theta_2 x_2}_{0} + \underbrace{\theta_3 x_1^2}_{1} + \underbrace{\theta_4 x_2^2}_{1})$

Predict "$y = 1$" if $\underbrace{-1 + x_1^2 + x_2^2}_{\geq 0}$

$x_1^2 + x_2^2 = 1$

$x_1^2 + x_2^2 \geq 1$



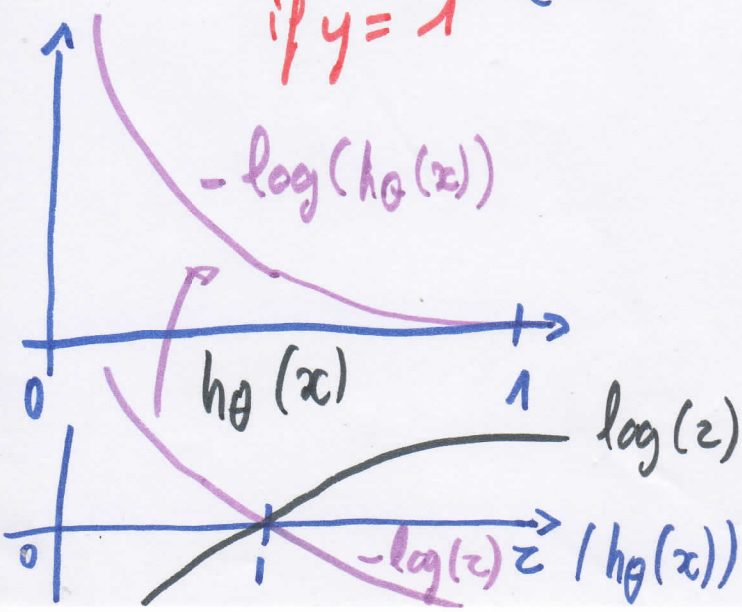
$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^3 x_2^2 \ldots)$

→ can be very complex

Logistic regression cost function

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

if $y = 1$



Cost = 0 if $y = 1, h_\theta(x) = 1$

But as $h_\theta(x) \to 0$

Cost $\to \infty$

Captures intuition that $h_\theta(x) = 0$ (predict $P(y = 1 | x; \theta) = 0$), but $y = 1$ we'll penalize learning algorithm by a very large cost

Andrew Ng - MACHINE LEARNING Stanford (16)

Logistic regression cost function

- if $h_\theta(x) = y$, then $\text{cost}(h_\theta(x), y) = 0$ (for $y=0$ and $y=1$)
  $\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$ but not convex so not working
- if $y=0$, then $\text{cost}(h_\theta(x), y) \to \infty$ as $h(x) \to 1$
- Regardless of whether $y=0$ or $y=1$ if $h_\theta(x) = 0.5$, then $\text{cost}(h_\theta(x), y) > 0$

Optimization algorithm

[Gradient descent]
- Conjugate gradient
- BFGS
- L-BFGS

Advantages
- No need to manually pick α
- Often faster than gradient descent

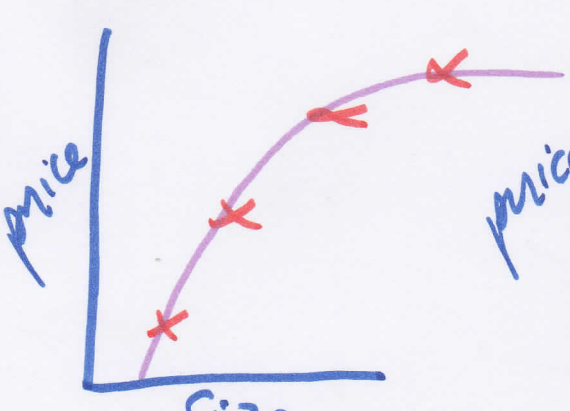
Disadvantages
- More complex

Underfitting & overfitting for linear regression

Example

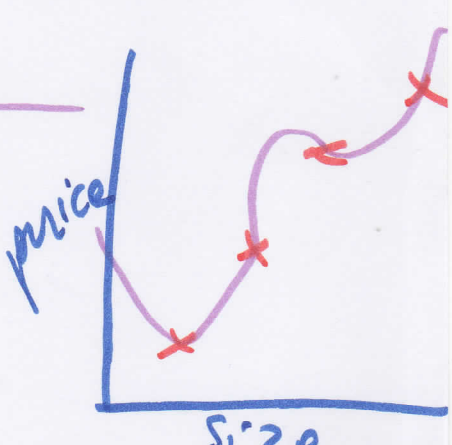
$\theta_0 + \theta_1 x$

"Underfit" "High bias" (not fitting)



$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"Overfit"
"High Variance"

Overfitting = if we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \simeq 0$), but fail to generalize to new example (predict price on new examples).

Underfitting & overfitting for logistic regression

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
(g = sigmoid function)
"Underfit"

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$
"Just right"

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1 x_2^2 + \theta_5 x_1^2 x_2 + \dots)$
"Overfit"

- When we have many features & not a lot of training examples, overfitting can become a problem.

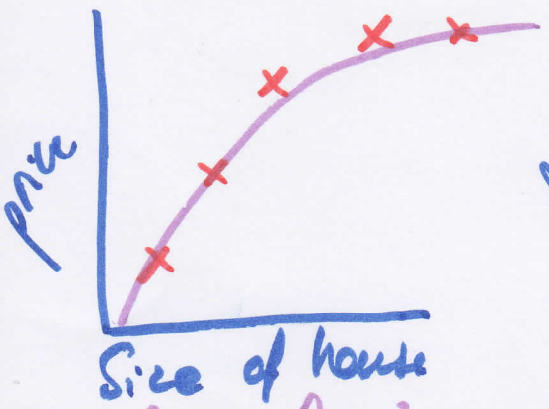
ex: x_1 = size of house, x_2 = nb. of bedrooms, x_3 = no. of floors...
... x_{100} ...

Addressing overfitting: Options:

1. Reduce no. of features
   - Manually select features to keep
   - Model selection algorithm
2. Regularization
   - Keep all the features, but reduce magnitude/values of θ_j
   - Works well when we have a lot of features, contribute predictive y

Regularization

- Small values for parameters $\theta_0, \theta_1, \dots \theta_n$
  - "Simpler" hypothesis
  - Less prone to overfitting
- Intuition



$\theta_0 + \theta_1 x + \theta_2 x^2$

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

fit better the data

$\theta_3 \simeq 0$ $\theta_4 \simeq 0$

Suppose we penalize θ_3, θ_4 and make them really small

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

with $\theta_3 \simeq 0$ $\theta_4 \simeq 0$

Regularized gradient descent

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right] \quad (j = 1, 2, 3, \dots, n)$$

$\}$

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$(1 - \alpha \frac{\lambda}{m}) < 1$

$\theta_1, \theta_2 \dots \theta_n$ are penalized. Not θ_0, that is why $\neq$ treatment